Ngôn Ngữ Lập Trình C/C++

(C/C++ Programming Language)

Trịnh Tấn Đạt
Khoa CNTT - Đại Học Sài Gòn
Email: trinhtandat@sgu.edu.vn
Website: https://sites.google.com/site/ttdat88/

Nội dung

- Các khái niệm cơ bản
- Các thành phần trong ngôn ngữ C/C++
- Các ví dụ minh họa
- Bài tập

Các khái niệm cơ bản

- Ngôn ngữ lập trình
- Chương trình dịch
- Soạn thảo mã nguồn - Biên dịch - Liên kết và Thực thi
- Giới thiệu ngôn ngữ lập trình C

Ngôn ngữ lập trình

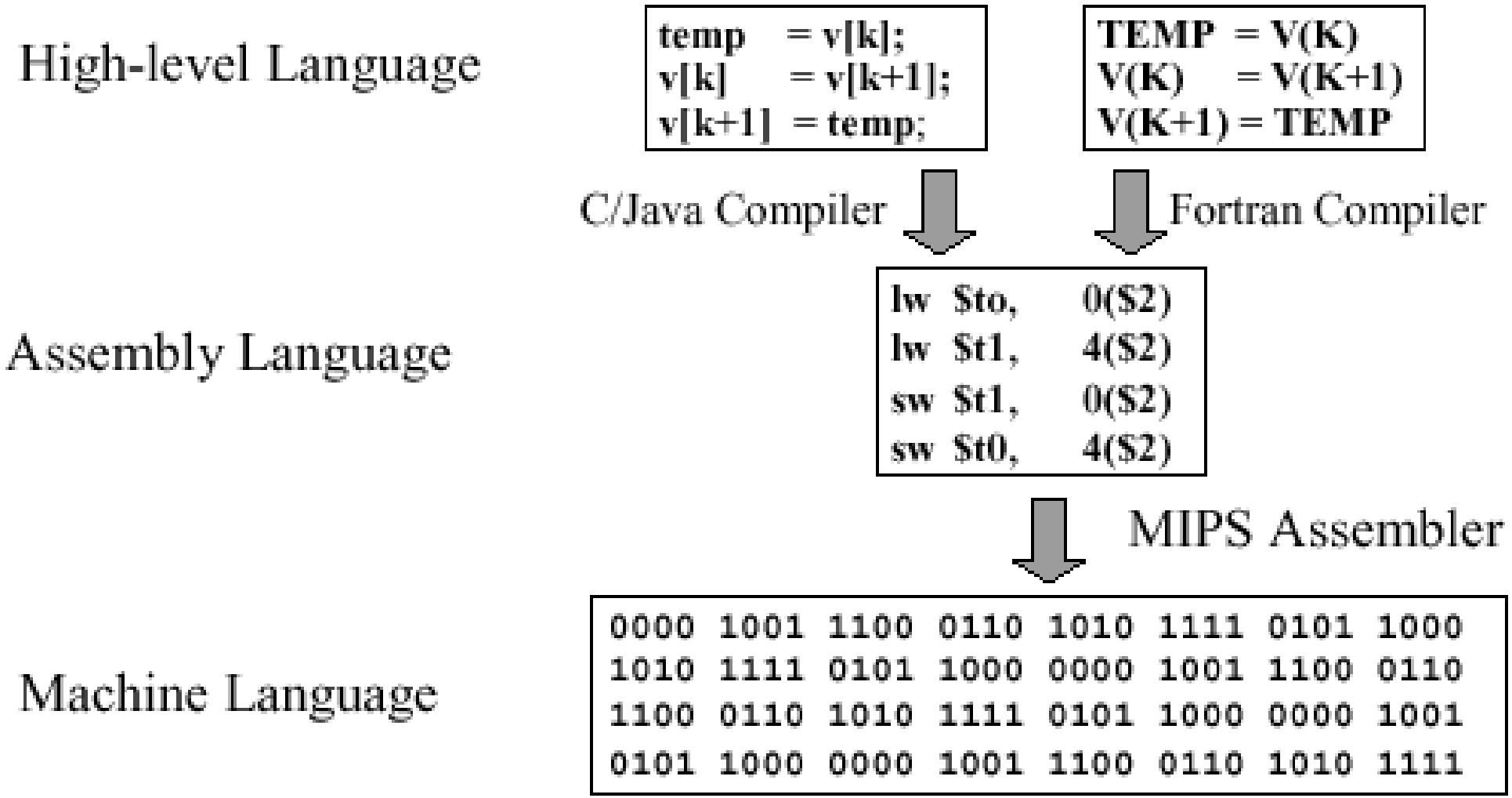
❖ Ngôn ngữ lập trình (**programming language**)

- Con người liên lạc với nhau dùng:
  - Ngôn ngữ tự nhiên: các mẫu từ ngữ và âm thanh
- Con người "nói chuyện" với máy tính dùng:
  - Ngôn ngữ lập trình: tập từ ngữ và ký hiệu
    - Tuân theo các luật được gọi là cú pháp (**syntax**)
- Có rất nhiều ngôn ngữ lập trình đang được sử dụng

Ngôn ngữ lập trình

- Dựa vào mức độ chi tiết hóa việc mô tả các thao tác, người ta chia ngôn ngữ lập trình thành các lớp:
  - **Ngôn ngữ máy** (machine code , low-level programming language)
  - **Hợp ngữ** (assembly)
  - **Ngôn ngữ cấp cao** (high-level programming language)
- Các ngôn ngữ cấp cao gần với ngôn ngữ tự nhiên nên rất tiện lợi cho việc mô tả các thao tác và dễ học, dễ nhớ.

Ngôn ngữ lập trình



Ngôn ngữ lập trình

❖ **Ngôn ngữ máy (machine code/language):**

- Là ngôn ngữ nền tảng của bộ vi xử lý, còn được gọi là mã máy.
- Tập lệnh của ngôn ngữ máy phụ thuộc vào loại vi xử lý (CPU) nên ngôn ngữ máy sẽ khác nhau trên những máy tính có sử dụng bộ vi xử lý khác nhau.
- Các chương trình được viết bằng các loại ngôn ngữ khác cuối cùng đều được chuyển thành ngôn ngữ máy trước khi chương trình đó được thi hành

Example of machine-language

Here's what a program-fragment looks like:

10100001 10111100 10010011 00000100
00001000 00000011 00000101 11000000
10010011 00000100 00001000 10100011
11000000 10010100 00000100 00001000

It means: z = x + y;

Ngôn ngữ lập trình

- Ưu điểm (**Pros**) viết chương trình bằng ngôn ngữ máy:
  - điều khiển máy tính trực tiếp
  - đạt được chính xác điều mình muốn làm.
  - hiệu quả về tốc độ thi hành, kích thước chương trình nhỏ

=> ngôn ngữ máy cho phép khai thác triệt để khả năng của máy tính.

- Khuyết điểm (**Cons**) của chương trình ngôn ngữ máy:
  - Tốn rất nhiều thời gian để viết,
  - Rất khó đọc, khó theo dõi để tìm lỗi
  - Chỉ chạy được trên những máy tính có cùng bộ vi xử lý.

=> ngôn ngữ máy được gọi là ngôn ngữ cấp thấp.

Ngôn ngữ lập trình

❖ **Hợp ngữ (assembly):**

- Tương tự như ngôn ngữ máy nhưng sử dụng các ký hiệu gợi nhớ để biểu diễn cho mã lệnh của máy.
- Được phát triển nhằm giúp lập trình viên dễ nhớ các lệnh của chương trình.
- Các chương trình hợp ngữ được chuyển sang mã máy thông qua trình hợp dịch (assembler).

Ví dụ: một đoạn assembly code

```
1    Load r1, X
2    Load r2, Y
3    Mult r3, r2, r1
4    Load r4, A
5    Mult r2, r4, r1
6    Add r5, r2, r4
7    Mult r1, r2, r5
8    Load r3, B
9    Add r7, r1, r3
```

Ngôn ngữ lập trình

❖ **Ngôn ngữ cấp cao (high-level programming language)**

- Hợp ngữ vẫn còn rất gần với từng thiết kế của máy tính.
- Cần có những ngôn ngữ lập trình gần với ngôn ngữ tự nhiên hơn, được gọi là ngôn ngữ cấp cao.
- Ngôn ngữ cấp cao bao gồm: danh từ, động từ, ký hiệu toán học, liên hệ và thao tác luận lý. Các yếu tố này có thể được liên kết với nhau tạo thành câu lệnh.
- Ưu điểm (Pros) viết chương trình bằng ngôn ngữ cấp cao:
  - Dễ đọc và dễ học
  - Không phụ thuộc vào máy tính

Ví dụ

- Một đoạn code ngôn ngữ lập trình cấp cao (ngôn ngữ C++)

```
#include <iostream>
using namespace std;
int main()
{
	cout<<"Hello World";
	cout<<endl;
	int a =1, b = 2;
	cout<<"tong a va b la:" << a+b;
	return 0;
}
```

Các thành phần của ngôn ngữ lập trình

- Mỗi ngôn ngữ lập trình thường có ba thành phần cơ bản:
  - **Bảng chữ cái:** là tập hợp các kí tự được dùng khi viết chương trình, ngoài các kí tự này không được phép dùng bất kì kí tự nào khác.
  - **Cú pháp:** là bộ quy tắc để viết chương trình. Dựa vào chúng, người lập trình và chương trình dịch biết được tổ hợp nào của các kí tự trong bảng chữ cái là hợp lệ và tổ hợp nào là không hợp lệ. Nhờ đó, có thể mô tả chính xác thuật toán để máy thực hiện.
  - **Ngữ nghĩa:** xác định ý nghĩa thao tác cần phải thực hiện, ứng với mỗi tổ hợp kí tự và dựa vào ngữ cảnh của nó.

Ví dụ:

- Hầu như các ngôn ngữ lập trình đều có **kí tự + chỉ phép cộng**. Xét các biểu thức:

  A + B (1)

  I + J (2)

- Giả thiết A, B là các biến thực và I, J là các biến nguyên.
- Khi đó dấu trong biểu thức (1) sẽ được hiểu là cộng hai số nguyên, dấu + trong biểu thức (2) sẽ được hiểu là cộng hai số thực.
- Như vậy, cú pháp cho biết cách viết một chương trình hợp lệ,
- Còn ngữ nghĩa xác định tính chất, thuộc tính của các tổ hợp kí tự trong chương trình.

Ví dụ

- Với **ngôn ngữ PHP** để hiển thị một thông báo ra về năm hiện tại màn hình chúng ta có thể viết như sau:

```
echo "Xin chào!";
```

- Đối với **ngôn ngữ Ruby** chúng ta lại sử dụng cú pháp khác:

```
puts("Xin chào!");
```

- Đối với **ngôn ngữ C**

```
printf("Xin chào!");
```

- Đối với **ngôn ngữ C++**

```
cout << "Xin chào!";
```

Chương trình dịch

- Chuyển đổi chương trình từ ngôn ngữ cấp cao (hợp ngữ) thành ngôn ngữ máy.
- Có hai kỹ thuật chính:
  - Trình biên dịch (**compiler**),
  - Trình thông dịch (**interpreter**).

Trình biên dịch

- Chuyển đổi toàn bộ chương trình sang ngôn ngữ máy và lưu kết quả vào đĩa để có thể thi hành về sau.
  - Chương trình nguồn (source program) là chương trình ngôn ngữ cấp cao được chuyển đổi.
  - Chương trình đối tượng (object program) là chương trình ngôn ngữ máy được tạo ra.
- Thực hiện
  - Duyệt, kiểm tra cú pháp chương trình,
  - Kiểm tra logic và đảm bảo các dữ liệu sử dụng được định nghĩa hợp lý,
  - Phát hiện và tạo ra một danh sách lỗi cú pháp của các mệnh đề (statement).
- Phương pháp dịch này thuận tiện cho các chương trình ổn định và cần thực hiện nhiều lần.

Trình thông dịch

- Lần lượt dịch và thực hiện từng câu lệnh một.
- Mỗi lần chạy chương trình là mỗi lần chương trình nguồn được thông dịch sang ngôn ngữ máy.
- Ưu điểm (Pros)
  - Có thể chạy một chương trình vẫn còn lỗi cú pháp.
- Nhược điểm (Cons)
  - Chậm hơn các chương trình được biên dịch.
- Phương pháp dịch này thích hợp cho môi trường đối thoại giữa người và hệ thống.

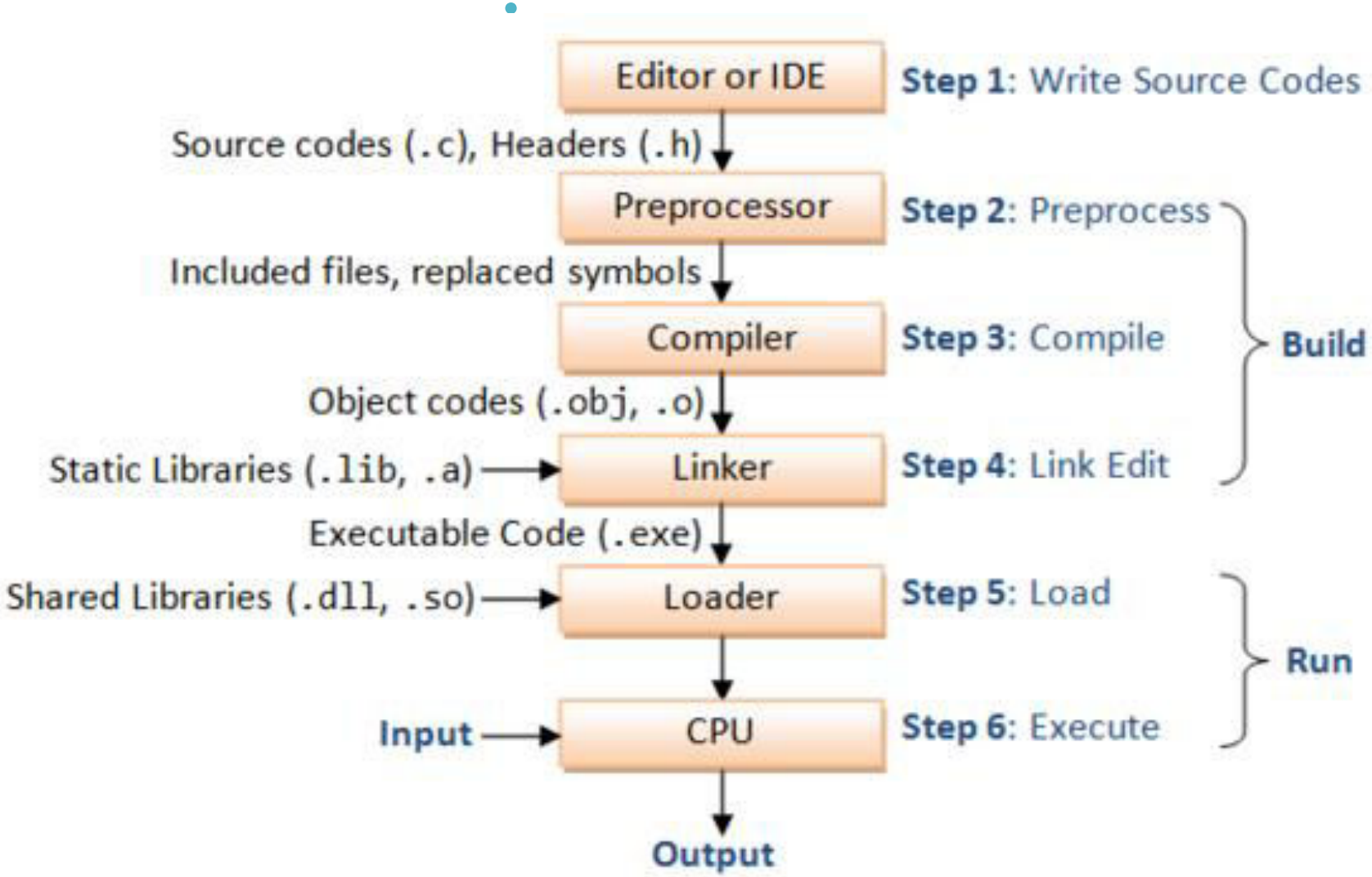
Soạn thảo mã nguồn – Biên dịch – Liên kết và Thực thi

- Mỗi ngôn ngữ lập trình có một vài môi trường lập trình tương ứng.
- Ví dụ ngôn ngữ C có các môi trường lập trình C-Free, Borland C, Dev C, Microsoft Visual Studio, …
- Môi trường lập trình cung cấp các dịch vụ như:
  - Soạn thảo mã nguồn,
  - Lưu trữ, tìm kiếm,
  - Xác định loại lỗi nếu có, chỉ rõ lỗi ở câu lệnh nào,
  - Cho xem các kết quả trung gian,
  - …

Soạn thảo mã nguồn – Biên dịch – Liên kết và Thực thi

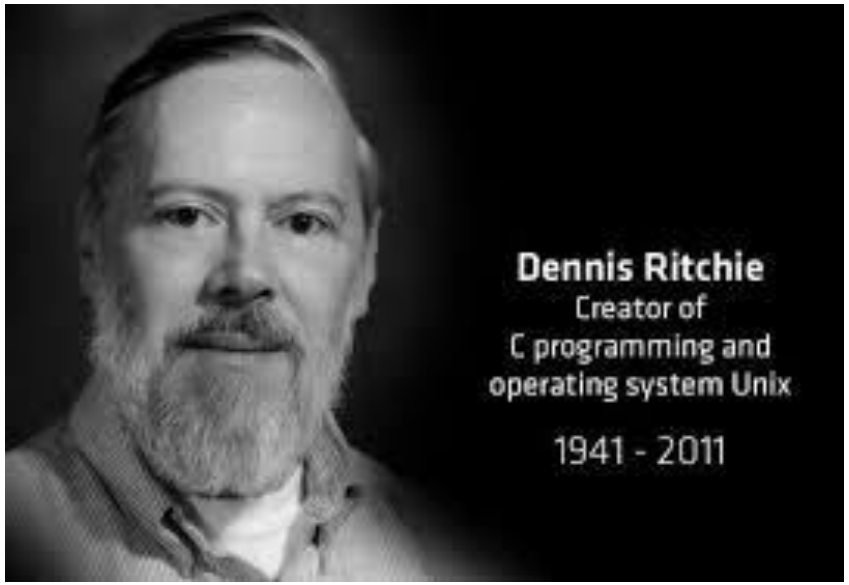
- Các môi trường lập trình khác biệt nhau ở các loại dịch vụ mà nó có thể cung cấp.
- Đặc biệt là các dịch vụ mở rộng, nâng cấp, tăng cường các khả năng mới cho ngôn ngữ lập trình.
- Khi biên dịch chương trình nguồn, người lập trình sẽ phát hiện được các lỗi cú pháp.
- Khi liên kết và thực thi chương trình trên dữ liệu cụ thể thì mới phát hiện được các lỗi ngữ nghĩa.

Soạn thảo mã nguồn – Biên dịch – Liên kết và Thực thi



Giới thiệu ngôn ngữ lập trình C

- Ngôn ngữ lập trình C được **Dennis Ritchie** xây dựng và phát triển từ ngôn ngữ B tại Bell Laboratories vào năm **1972**.
- Khởi đầu ngôn ngữ lập trình C được sử dụng rộng dãi trong hệ điều hành UNIX.
- Ngày nay, các chương trình viết bằng ngôn ngữ C có thể chạy trên hầu hết các hệ điều hành



Giới thiệu ngôn ngữ lập trình C

- Điểm mạnh ngôn ngữ C:
  - Rất mạnh và linh động, có khả năng thể hiện bất cứ ý tưởng nào.
  - Được sử dụng rộng rãi bởi các nhà lập trình chuyên nghiệp.
  - Có tính khả chuyển, ít thay đổi trên các hệ thống máy tính khác nhau.
  - Rõ ràng, cô đọng.
  - Lập trình đơn thể, tái sử dụng thông qua hàm.

Giới thiệu ngôn ngữ lập trình C

- Điểm yếu ngôn ngữ C:
  - Dễ bị lỗi: các lỗi chỉ được phát hiện khi thực hiện biên dịch chương trình.
  - Có thể khó hiểu: từ khóa và cú pháp phức tạp
  - Khó chỉnh sửa: Các chương trình lớn viết bằng ngôn ngữ C có thể khó chỉnh sửa nếu các tài liệu thiết kế chương trình không được thực hiện chi tiết.

Phân Biệt **C** và **C++**.

- **C** là ngôn ngữ ra đời trước, và là ngôn ngữ hướng thủ tục, nó dễ dàng được triển khai và chạy trên các hệ điều hành.
- **C++** ra đời sau mở rộng từ **C** nó mang vào khái niệm lập trình hướng đối tượng, **C** là nền tảng của **C++**, và **C++** không ra đời để thay thế **C**, các thư viện của nó được mở rộng lên rất nhiều.

Ví dụ một chương trình C/C++

Chương trình C

```c
#include <stdio.h>
#include <conio.h>

int main()
{
    int x, y, tong;
    printf("Nhap hai so nguyen: ");
    scanf("%d%d", &x, &y);
    tong = x + y;
    printf("Tong hai so la %d", tong);
    return 0;
}
```

Chương trình C++

```cpp
#include <iostream>
using namespace std;
int main()
{
    int x, y, tong;
    cout<<"Nhap hai so nguyen: ";
    cin >> x >> y;
    tong = x + y;
    cout<< "tong hai so la: " << tong;
    return 0;
}
```

Cấu trúc chương trình C/C++ đơn giản

```
#include "…";  // Khai báo file tiêu đề, thư viện

int x;     // Khai báo biến hàm
void Nhap(); // Khai báo hàm

int main()// Hàm chính
{
   // Các lệnh và thủ tục
}

void Nhap()    // Định nghĩa hàm
{
    // Chi tiết trong hàm
}
```

Cấu trúc chương trình C/C++ đơn giản

```
1  #include <stdio.h>
2
3  void main()
4  {
5      // in ra màn hình
6       printf("Hello World");
7  }
```

✓ Dòng đầu tiên : chứa phát biểu tiền xử lý

#include <stdio.h>

- ***Trong chương trình này ta*** sử dụng hàm thư viện của C là ***printf, do đó bạn cần phải có khai báo của hàm thư viện này để*** báo cho trình biên dịch C biết. **Nếu không khai báo chương trình sẽ báo lỗi.**

VD: iostream.h → cout, cin, ...

stdio.h → printf, scanf, ...

math.h → sqrt, sin, log, pow, ...

✓ Dòng thứ 2: hàng trắng viết ra với ý đồ làm cho bảng chương trình thoáng, dễ đọc.

Cấu trúc chương trình C/C++ đơn giản

```
1  #include <stdio.h>
2
3  void main()
4  {
5      // in ra màn hình
6       printf("Hello World");
7  }
```

- ✓ Dòng thứ 3: **void main()** ***là thành phần chính của mọi chương trình C*** *(bạn có thể viết* main() hoặc void main() hoặc void main(void)). **Bắt buộc phải có** trong mọi chương trình C/C++.
- Cặp dấu ngoặc () cho biết đây là hàm (function).
- Hàm void main() có từ khóa **void** cho biết hàm này không trả về giá trị.
- Ngoài ra, có thể dùng

```
int main()
{
        return 0;
 }
```

Cấu trúc chương trình C/C++ đơn giản

```
1  #include <stdio.h>
2
3  void main()
4  {
5      // in ra màn hình
6       printf("Hello World");
7  }
```

- ✓ Dòng thứ 4 và 7: cặp dấu ngoặc móc { } giới hạn **thân hàm**. Thân hàm bắt đầu bằng dấu { và kết thúc bằng dấu }.
- ✓ Dòng 6: Trong thân hàm gồm định nghĩa dữ liệu và các phát biểu
- Các phát biểu là phần thực thi của chương trình. (đọc từ bàn phím, xuất ra màn hình, thực hiện tính toán, gọi hàm,...)
- Các phát biểu được đặt giữa cặp ngoặc { và } của hàm (main), tạo nên "thân hàm".
- Mỗi phát biểu đơn (câu lệnh) được **kết thúc bởi dấu chấm phẩy (;)**
- Các phát biểu cùng được đặt giữa { và } tạo thành phát biểu ghép (còn gọi khối lệnh).
- Hàm **printf("Hello World");**, sẽ in chữ Hello World ra màn hình

Cấu trúc chương trình C/C++ đơn giản

```
1  #include <stdio.h>
2
3  void main()
4  {
5      // in ra màn hình
6       printf("Hello World");
7  }
```

✓ Dòng thứ 5: Các chú thích trong lúc lập trình (code)

- **Các chú thích, được trình biên dịch "bỏ qua", KHÔNG** ảnh hưởng đến việc thực thi của chương trình.
- *Có hai loại chú thích:*
  - *Chú thích khối, chú thích là phần văn bản đặt giữa* /* *và* */ .

Ví dụ :

```
/*
        ghi chú thích
*/
```

  - *Chú thích dòng, chú thích là phần văn bản đặt ngay sau cặp kí* tự: //

Lưu ý: Một số qui tắc viết chương trình C

- Mỗi câu lệnh có thể viết trên một hay nhiều dòng nhưng phải được **kết thúc bằng dấu (;).**
- Một chương trình **chỉ có một hàm chính** (main), có thể có thêm vài hàm khác (gọi là hàm con).
  - **Kết thúc tên hàm không có dấu chấm phẩy hoặc bất cứ dấu gì.**
  - **Thân hàm phải được bao bởi cặp { }.**
  - **Các câu lệnh trong thân hàm phải viết thụt vào (canh lề).**
- Khi sử dụng một hàm thư viện cần khai báo hàm ở đầu chương trình bằng cách từ khóa, **#include** ví dụ: **#include<iostream.h>**
- Chú thích có thể được viết trên một dòng, nhiều dòng hoặc trên phần còn lại của câu lệnh.
- Các từ **include, stdio.h, void, main, printf, int** phải viết **bằng chữ thường**.
- Chuỗi trong nháy kép cần in ra "Bạn có thể viết chữ HOA, thường tùy, ý".

Ôn Tập

1. Ngôn ngữ lập trình là gì? Trình bày các ưu là khuyết điểm của các 3 lớp ngôn ngữ lập trình (machine code, assembly, high-level)
2. Trình bày cấu trúc của một chương trình C. Giải thích ý nghĩa của từng phần. Cho ví dụ minh họa.
3. Câu ghi chú (chú thích) dùng để làm gì? Cách sử dụng ra sao? Cho ví dụ minh họa.

Các thành phần trong C/C++

- Bộ từ vựng, Tên, Từ khóa
- Các kiểu dữ liệu cơ sở
- Biến và hằng
- Phép toán, biểu thức
- Lệnh, khối lệnh
- Lệnh xuất/ nhập
- Các ví dụ minh họa

Bộ từ vựng trong C/C++

❖ Các ký tự được sử dụng

- Bộ chữ cái 26 ký tự Latinh **A, B, C, …, Z, a, b, c, …, z**
- Bộ chữ số thập phân : **0, 1, 2, …, 9**
- Các ký hiệu toán học : **+ – * / = < > ( )**
- Các ký tự đặc biệt : **. , : ; [ ] % \ # $ ‘**
- Ký tự gạch nối _ và khoảng trắng ' '

Tên/ Định danh

❖ Tên/Định danh (**Identifier**)

- Một dãy ký tự dùng để chỉ tên một hằng số, hằng ký tự, tên một biến, một kiểu dữ liệu, một hàm một hay thủ tục. Gồm :
  - Ký tự chữ
  - Ký tự số
  - Ký tự '_' (underscore character).
- Qui tắc khi đặt tên:
  - **KHÔNG** được trùng với các **từ khóa**
  - Bắt buộc chữ đầu phải là chữ cái hoặc _. **KHÔNG** bắt đầu bằng chữ số.
  - Số ký tự tối đa trong một tên là 255 ký tự và được dùng ký tự _ chen trong tên.
  - **KHÔNG** cho phép chen giữa các ký tự bằng **khoảng trắng.**
  - Phân biệt chữ **IN HOA** và **in thường.**

Tên/ Định danh

Ví dụ: Các tên sau đây

- Các tên hợp lệ: GiaiPhuongTrinh, Bai_Tap1, _diemthi, flag
- Các tên không hợp lệ: 1A, Giai Phuong Trinh, main, include
- Các tên sau đây khác nhau:
  - A, a
  - BaiTap, baitap, BAITAP, bAItaP, …

Từ khóa

❖ Từ khóa (**Keyword**)

- Các "tên" dành riêng trong ngôn ngữ dùng cho các mục đích khác nhau.
- KHÔNG thể sử dụng **từ khóa** để đặt tên cho biến, hàm, tên chương trình con.
- Một số từ khóa thông dụng:
  - const, enum, signed, struct, typedef, unsigned…
  - char, double, float, int, long, short, void
  - case, default, else, if, switch
  - do, for, while
  - break, continue, return

Câu hỏi

1. Tên (định danh) nào sau đây đặt không hợp lệ, tại sao?

a) Tin hoc co SO A

b) 1BaiTapKHO

c) trungBinhcOng

d) CoSo_L@pTrinh

e) _Tinhoc

e) double

g) return

h) temp

Kiểu dữ liệu

- Xét tập **N, Z, Q, R, C ?!**
- Kiểu dữ liệu (KDL - **data type**) được xác định bởi:
  - *tập giá trị, và*
  - *tập các phép toán tác động lên các phần tử thuộc tập* giá trị ấy.
- Đơn vị lưu trữ là *byte*. *Mỗi giá trị thuộc một KDL* được biểu diễn bởi một số byte nhất định.

=> Các giá trị biểu diễn được là hữu hạn.

Ví dụ:

C Basic Data Types	32-bit CPU		64-bit CPU	
	Size (bytes)	Range	Size (bytes)	Range
char	1	-128 to 127	1	-128 to 127
short	2	-32,768 to 32,767	2	-32,768 to 32,767
int	4	-2,147,483,648 to 2,147,483,647	4	-2,147,483,648 to 2,147,483,647
long	4	-2,147,483,648 to 2,147,483,647	8	- 9,223,372,036,854,775,808- 9,223,372,036,854,775,807
long long	8	9,223,372,036,854,775,808- 9,223,372,036,854,775,807	8	9,223,372,036,854,775,808- 9,223,372,036,854,775,807
float	4	3.4E +/- 38	4	3.4E +/- 38
double	8	1.7E +/- 308	8	1.7E +/- 308

Kiểu dữ liệu cơ sở

❖ Có 4 kiểu cơ sở như sau:

- Kiểu số nguyên (**integer**): giá trị của nó là các số nguyên như 10, -20, 100, …
- Kiểu số thực (**float**): giá trị của nó là các số thực như 3.1415, 0.5, -1.01326, …
- Kiểu luận lý (**logic**): giá trị đúng hoặc sai.
- Kiểu ký tự (**character**): 256 ký tự trong bảng mã ASCII.

Kiểu số nguyên (**integer**)

❖ Các kiểu số nguyên (có dấu) và (không dấu)

- Kiểu **số nguyên có dấu** n bit, miền giá trị $-2^{n-1} \ldots +2^{n-1} - 1$

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
char	1	–128 ... +127
short	2	–32.768 ... +32.767
int	4	–2.147.483.648 ... +2.147.483.647
long	4	–2.147.483.648 ... +2.147.483.647

Kiểu số nguyên (integer)

- Kiểu **số nguyên không dấu** n bit, miền giá trị $0 \ldots 2^n - 1$
- Thêm tiếp đầu ngữ ***unsigned*** với kiểu số nguyên

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
unsigned char	1	0 … 255
unsigned short	2	0 … 65.535
unsigned int	4	0 … 4.294.967.295
unsigned long	4	0 … 4.294.967.295

Xem thêm các kiểu dữ liệu số nguyên khác: https://docs.microsoft.com/en-us/cpp/cpp/data-type-ranges?view=vs-2019

Kiểu số nguyên (integer)

- Lưu ý khi làm việc với số nguyên: Các phép toán trên số nguyên: + – * / %

8/4 → 2

9/4 → 2 (why???)

2/2 → 1

1/2 → 0 (why???)

9%5 → 4

Kiểu số thực (**float**)

- Các kiểu số thực (floating-point)

Kiểu (Type)	Độ lớn (Byte)	Miền giá trị (Range)
float	4	$3.4*10^{-38}$ ... $3.4*10^{38}$
double	8	$1.7*10^{-308}$ ... $1.7*10^{308}$

Lưu ý : Các phép toán trên số thực:

\+ – * /

⇒ Kiểu double được lưu ý sử dụng:

- Tính toán với số lớn.
- Cần độ chính xác cao

  - float: Độ chính xác đơn (Single-precision) chính xác đến 7 số lẻ.
  - double: Độ chính xác kép (Double-precision) chính xác đến 19 số lẻ.

- Hai cách biểu diễn số thực:
  - Dạng thập phân: phần nguyên & phần phân: 12.345 -0.02468
  - Dạng chấm động: phần định trị & phần mũ: 1.2345e+01 -2.468e-02

Kiểu ký tự (character)

- Tên kiểu: **char**
- Biểu diễn hằng kí tự: 'a', '4', '@',...
- Tập giá trị (1 byte, mã hoá được 256 kí tự):
  - Kí tự chữ ('a', 'S',...)
  - Kí tự số ('0',..,'9')
  - Dấu ('@','?',..)
  - Kí tự điều khiển ('\n', '\t',...)
  - Kí tự đặc biệt.

Lưu ý: các ký tự phải đặt trong dấu nhấy đơn

Ví dụ: trong bảng mã ASCII
Lưu số 65 tương đương với ký tự 'A'…
Lưu số 97 tương đương với ký tự 'a'.

- Chính là kiểu số nguyên do:
  - Mỗi kí tự được lưu với một số nguyên, và theo một thứ tự nhất định gọi là bộ mã.
  - Bộ mã được dùng phổ biến là bộ mã ASCII.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Kiểu ký tự

- Các phép toán trên kiểu ký tự tương tự số nguyên

+ – * / %

- Thực hiện trên mã ASCII của kí tự tương ứng:

c = 'A' //c= 65

c+1 → 66 → 'B'

'a' – 'A' → 32

'8' – '3' → 5

Kiểu luận lý (boolean)

- Trong C ngầm định một cách không tường minh:
  - false (sai): giá trị 0.
  - true (đúng): giá trị khác 0, thường là 1.
- Trong C++: bool
- **Ví dụ**

0 (false), 1 (true), 2 (true), 2.5 (true)

1 > 2 (0, false), 1 < 2 (1, true)

Định nghĩa kiểu với typedef

- Một khai báo có thêm tiền tố ***typedef sẽ định nghĩa*** một tên mới cho kiểu dữ liệu (đã có).

typedef kieudulieu tenMoi;

- Một tên được định nghĩa theo cách này được gọi là *"định nghĩa kiểu"*.

Ví dụ:

```
typedef   long  SoNg64;
typedef    int   SoNg32;
typedef    char  KYTU;
```

Biến (Variable)

- Định nghĩa biến (khai báo biến) là ***đặt tên*** và xác định ***kiểu dữ liệu*** cho biến.
- Mọi biến cần phải được khai báo trong chương trình trước khi sử dụng.
- Để định nghĩa **một** biến, dạng khai báo (cú pháp)

  <data type> <tenBien>; *(lưu ý dấu chấm phẩy ở cuối)*

- Định nghĩa **nhiều** biến **cùng kiểu**:

  <data type> <tenBien1>, <tenBien2>, <tenBienN>;

Ví dụ:

```
int i;                  // i là biến số nguyên
int j, k;               // j và k là 2 biến số nguyên
unsigned char dem;      // dem là kiểu ký tự không dấu
float ketqua, delta;    // ketqua và delta là 2 biến kiểu float
```

Biến (Variable)

❖ Gán giá trị cho biến (**Variable assignment**)

- Phép gán "**=**" **để thay đổi giá trị biến**

tenBien = giatri;

Ví dụ: int a ; float b;

a = 10; b = 0.55;

- Gán liên tiếp là gán cho nhiều biến cùng lúc sau khi đã khai báo các biến.

Ví dụ: int a, b ;

a = b = 100;

Biến (Variable)

❖ Gán giá trị cho biến (variable assignment)

- Khởi tạo giá trị cho biến (**variable initialization**): là gán trị cho biến ngay khi khai báo biến đó.

Ví dụ:

```
double  x = 3.14;
int y = 5;
char  c = 'A';
```

Biến (Variable)

- Gán kép và khởi tạo: int a = b = 6;
- Chú ý phân biệt:

```
double x= 1.0, y= 2.0, z= 1.5;
int a, b;       // khai báo biến
a = b = 6;    // gán giá trị cho  biến
int a = b = 6;   // khởi tạo giá trị cho biến
int m = 3, n = 3;
```

Hằng (constant)

- Hắng số (**constant**): là đại lượng không đổi trong suốt quá trình thực thi của chương trình.
- Phân biệt:
  - Hằng biến
  - Hằng thực sự
  - Hằng ký hiệu
- Định nghĩa hằng dùng từ khóa: **const, define**

Hằng (constant)

- Định nghĩa dùng từ khóa: **const**
- Cú pháp : **const <data type> <TENHANG> = <giatricuthe>;**
- **Lưu ý:** thông thường TÊN HẰNG được viết IN HOA

Ví dụ:

```
const float PI = 3.14;
const int MAX = 1000;
```

Hằng (constant)

- Định nghĩa bằng từ khóa: **define**
- Cú pháp : **#define <TENHANG> <giatri>**
- **Lưu ý:**
  - KHÔNG có dấu chấm phẩy sau <giatri>
  - KHÔNG dùng phép gán =
  - Một định nghĩa chỉ một hằng

Ví dụ :

```
#define  PI  3.14
#define MIN 0
```

- #define: Đây là tiền xử lý (preprocessors) hay gọi là các **marco** definition.
- **Lưu ý:** Khi dùng **#define** để định nghĩa hằng. Phải đặt câu lệnh này ở phần đầu chương trình (chung phần khai báo thư viện **#include**, bên trên các hàm). Vì đây là các tiền xử lý.

Hằng ký tự và Hằng chuỗi

❖ Hằng ký tự và hằng chuỗi

- Hằng ký tự: 'A', 'a', … (ký tự đặt trong dấu nhấy đơn)

Ví dụ: char c = 'B';

- Hằng chuỗi: "Hello World!", "Nguyen Van A" (chuỗi đặt trong dấu nháy kép)

Ví dụ : char *s = "Hello World!";

- Chú ý: 'A' khác "A" .

Ví dụ:

Dùng const

```
// Tính dien tích và chu vi hình tròn
#include<iostream.h>
#include<math.h>
void main()
{
    const double PI = 4*atan(1);
    double R, S, C;
    cout<<"Ban hay nhap ban kinh hinh tron: ";
    cin>>R;
    if (R<=0)
        R= 0;
    S= PI*R*R;
    C= 2*PI*R;
    cout.precision(10);                  //chu y
    cout.flags(ios::scientific);         //chu y
    cout<<" Dien tich hinh tron: "<<S;
    cout<<"\n Chu vi duong tron: "<<C;
}
```

Dùng #define

```
#include<iostream.h>

#define PI 3.14159
void main()
{
    double R, S, C;
    cout<<"Moi ban nhap ban kinh hinh tron:";
    cin>>R;
    if (R<0)
        R= 0;
    C= 2*PI*R;      S= PI*R*R;
    cout<<"Chu vi duong tron:"<<C<<"\n";
    cout<<"Dien tich hinh tron:"<<S<<"\n";
}
```

Biểu thức

- Biểu thức (**expression**) : là sự kết hợp hợp lệ giữa các toán hạng (**operand**) và toán tử (**operator**) để diễn đạt một công thức toán học nào đó, cho một kết quả duy nhất sau cùng.
  - Toán tử: +, –, *, /, %, <, >….
  - Toán hạng: hằng, biến, lời gọi hàm...

  Ví dụ: delta= b*b – 4*a*c;

  pi= 4*atan(1.0);
- Biểu thức với toán tử là phép toán số học → biểu thức số học
- Biểu thức với phép toán quan hệ & luận lí → biểu thức quan hệ & luận lí.

Phép toán

- Phép toán hay còn gọi là **toán tử** (**operator**) : Trong C, các phép toán có thể phân ra thành 3 loại chính: phép toán số học, phép thao tác bit, phép toán quan hệ và luận lý.
- Phép toán 1 ngôi (**unary** operator), còn gọi là phép toán 1 toán hạng. Ví dụ: 5! , log(2.5), sin(2*PI)
- Phép toán 2 ngôi (**binary** operator), còn gọi là phép toán 2 toán hạng. Ví dụ: 2+5 , 10/5
- Độ ưu tiên của phép toán qui định trình tự tính toán trong biểu thức.

Ví dụ: a = - 9/2*2 - 2 – 7%5;

Phép toán số học (Arithmetic operators)

- Các phép toán số học 1 ngôi: **+ -**
- Các phép toán số học 2 ngôi: *** / % + -**
- Phép chia nguyên và chia không nguyên: **/**

  Ví dụ: 11/2 = 5 11/2.0 = 5.5
- Phép toán % cho phần dư của phép chia nguyên.
- **Lưu ý:** Phép toán % không áp dụng được cho các giá trị kiểu float và double.

Phép toán số học

- **Ví dụ:**

a = 1 + 2; b = 1 – 2; c = 1 * 2; d = 1 / 2;

e = 1*1.0 / 2; f = float(1) / 2; g = float(1 / 2);

h = 1 % 2;

x = x * (2 + 3*5); ⇔ x *= 2 + 3*5;

x = x + y ⇔ x += y

Phép toán quan hệ (Relational operators)

- Phép toán quan hệ **dùng để so sanh 02 biểu thức** với nhau: > , <, <=, >=, == , !=
- Lưu ý: phép so sánh bằng == (hai dấu bằng) // phân biệt với phép gán =
- Phép toán quan hệ cho ta hoặc giá trị đúng (true hay 1) hoặc giá trị sai (false hay 0).

Ví dụ:

```
if (a>b)
        cout<<a<<"la so lon hon !";
if (b!=0)
        cout<<a/b;
```

Ví dụ:

```
s1 = (1 == 2);      s2 = (1 != 2);
s3 = (1 > 2);       s4 = (1 >= 2);
s5 = (1 < 2);       s6 = (1 <= 2);
```

- Các phép toán quan hệ có độ ưu tiên thấp hơn so với các phép toán số học

Phép toán luận lí (Logical operators)

- Phép toán luận lí tổ hợp **nhiều biểu thức quan hệ với nhau**: **&&** (and), **||** (or) , **!** (not)
- Phép toán luận lý cho ta hoặc giá trị đúng (true hay 1) hoặc giá trị sai (false hay 0).

  Ví dụ: 3 && 7 có giá trị 1
- Các phép toán quan hệ và luận lí được dùng để thiết lập điều kiện rẽ nhánh trong toán tử if và điều kiện kết thúc chu trình trong các toán tử for, while và do-while.

&&	0	1
0	0	0
1	0	1

||	0	1
0	0	1
1	1	1

Ví dụ:

```
s1 = (1 > 2) && (3 > 4);
s2 = (1 > 2) || (3 > 4);
s3 = !(1 > 2);
```

Phép toán trên bit

❖ Toán tử trên bit (**bitwise operator**) : Tác động lên các bit của toán hạng (nguyên).

- Các toán tử:

  & (and), | (or), ^ (xor), ~ (not hay lấy số bù 1)

  >> (shift right), << (shift left)

&	0	1
0	0	0
1	0	1

|	0	1
0	0	1
1	1	1

^	0	1
0	0	1
1	1	0

~	0	1
	1	0

Ví dụ:

```
    0110 (số thập phân 6)
AND 1101 (số thập phân 13)
  = 0100 (số thập phân 4)
Trong C viết :
 int a= 6, b = 13;
 int z = a & b;
```

```
NOT 0111 (số thập phân 7)
  = 1000 (số thập phân 8)
Trong C viết :
  int a = 7;
  int z = ~a;  // tại sao ra  -8
```

```
    0101 (số thập phân 5)
OR  0011 (số thập phân 3)
  = 0111 (số thập phân 7)
Trong C viết :
     int   a =  5, b = 3;
     int    z =  a | b;
```

```
     0101 (số thập phân 5)
XOR  0011 (số thập phân 3)
     = 0110 (số thập phân 6)
Trong C viết:
     int a = 5, b = 3;
     int z = a ^ b;
```

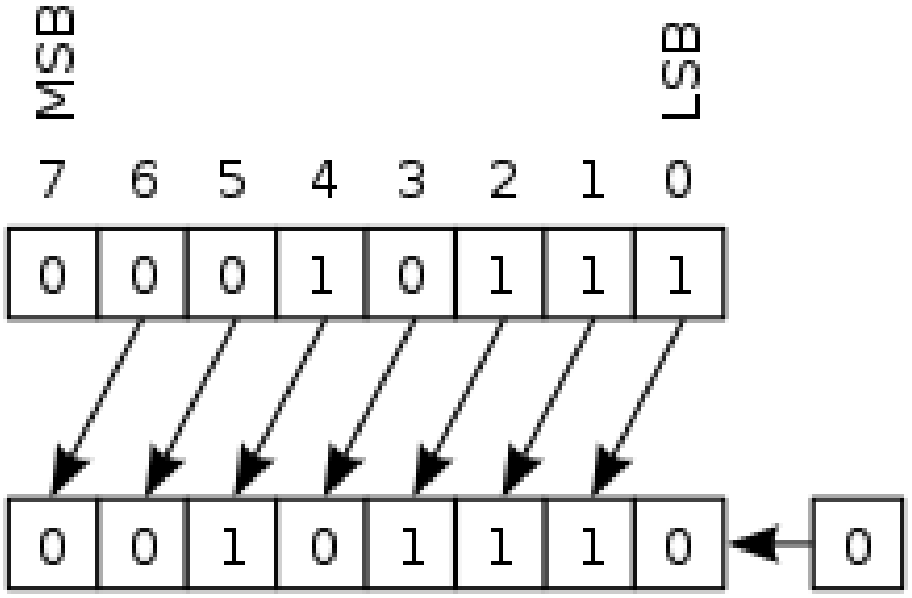
Ví dụ

- Dịch chuyển bit sang trái

00010111 (số thập phân +23) Dịch chuyển trái
= 00101110 (số thập phân +46)
Trong C viết:

```
int  a = 23;
int z1 = a <<1;  // ket qua 46
int z2 = a<<2;   // ket qua 92
int z3  = a<<3;   // ket qua 184
```

Câu hỏi: int y = a<<n; /// ket la bao nhieu



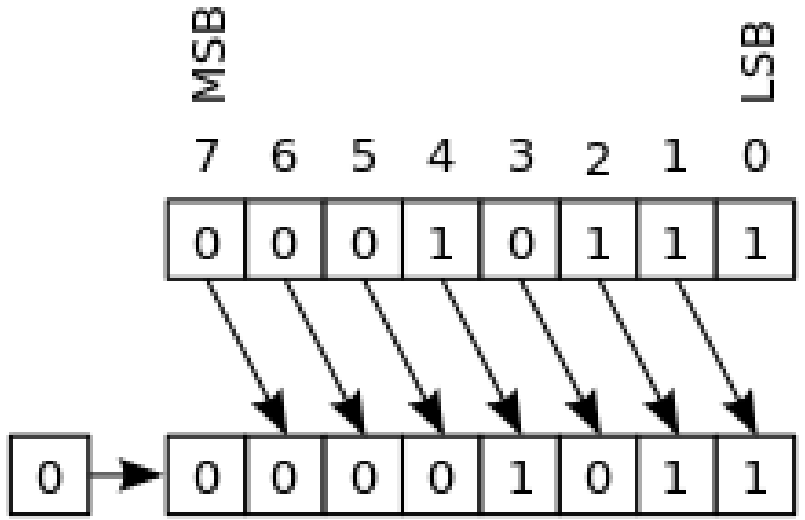
Ví dụ

- Dịch chuyển bit sang phải

00010111 (số thập phân +23) Dịch chuyển phải
= 00001011 (số thập phân +11)
Trong C viết:

```
int  a = 23;
int z1 = a >>1;  // ket qua 11
int z2 = a >>2;   // ket qua 5
int z3  = a >>3;   // ket qua 2
```

Câu hỏi: int y = a >> n; /// ket la bao nhieu



Ví dụ

```
void main()
{
    int a = 5;  // 0000 0000 0000 0101
    int b = 6;  // 0000 0000 0000 0110

    int z1, z2, z3, z4, z5, z6;
    z1 = a & b; // 0000 0000 0000 0100
    z2 = a | b; // 0000 0000 0000 0111
    z3 = a ^ b; // 0000 0000 0000 0011
    z4 = ~a; // 1111 1111 1111 1010
    z5 = a >> 2;// 0000 0000 0000 0001
    z6 = a << 2;// 0000 0000 0001 0100
}
```

Phép toán điều kiện

❖ Phép toán điều kiện (**Conditional** or **Ternary** Operator)

- Đây là toán tử 3 ngôi (gồm có 3 toán hạng)
- Cú pháp : <biểu thức 1> ? <biểu thức 2> : <biểu thức 3>
  - <biểu thức 1> đúng thì giá trị là <biểu thức 2>.
  - <biểu thức 1> sai thì giá trị là <biểu thức 3>.

Ví dụ

```
s1 = (1 > 2) ? 2912 : 1706;
int s2 = 0;
1 < 2 ? s2 = 2912 : s2 = 1706;
```

Kiểu bool (Boolean)

- Kiểu ***bool được dùng để biểu diễn kết quả của*** biểu thức luận lí, cho kết quả là ***đúng (true) hoặc sai (false).***
- Ngôn ngữ C không định nghĩa tường minh kiểu bool, được dùng thông qua *kiểu số nguyên.*
  - Kết quả biểu thức là ***true ⇒ giá trị là 1***
  - Kết quả biểu thức là ***false ⇒ giá trị là 0***

Kiểu bool

int a, b, c;

cin>>a>>b; // nhập giá trị a và b từ bàn phím

c= a>b; //c= 0 or 1

- Giá trị biểu thức là ≠ **0 ⇒ KQ ứng là *true***
- Giá trị biểu thức là **= 0 ⇒ KQ ứng là *false***

Ví dụ:

```
if (b)   // tương đương viết b == true
        cout<<a/b;
```

Câu hỏi

- Viết biểu thức cho các yêu cầu sau:

1. Kiểm tra một năm ***y có phải là năm nhuận ? Biết năm là*** *nhuận nếu là năm chia hết cho 400* ***hoặc*** *chia hết cho 4* ***nhưng*** *không chia hết cho 100.*
2. Kiểm tra a, b, c có thể là 3 cạnh của một tam giác ? *Tổng chiều dài của hai cạnh bất kì luôn lớn hơn chiều dài cạnh còn lại.*

Phép tăng và giảm

- Nếu phép ***tăng*** (**increment** operator, ký hiệu **++**) *đặt ngay* ***trước tên biến, là*** phép toán "*tăng trước*".
- Tương tự, phép ***giảm*** (**decrement** operator, ký hiệu **--**) *đặt ngay* ***trước tên biến****, là phép toán "giảm trước".*
- Giá trị của biến được *tăng (giảm) 1 đơn vị, sau đó giá* trị mới này được ***dùng trong biểu thức mà biến xuất*** hiện.

```
int a= 5, b= 6, c, d;
c= ++a + b;   // kq: c =12,  a tăng thành 6, trước khi cộng b
d= a * --b;    // kq: d = 30, b giảm thành 5 trước khi nhân a=6
```

Phép tăng và giảm

- Nếu phép ***tăng (++) đặt ngay sau tên biến, là phép toán*** "*tăng sau*".
- Nếu phép ***giảm (--) đặt ngay sau tên biến, là phép toán*** "*giảm sau*".
- Giá trị hiện tại của biến được ***dùng trong biểu thức mà*** nó xuất hiện, sau đó giá trị của biến mới được ***tăng (giảm) 1 đơn vị.***

```
int a= 5, b= 6, c, d;
c= a++ + b;  // kq: c= 11 , sau đó a tăng thành 6, a = 6
d= a * b--;  // kq:  d = 36 , sau đó b giảm thành 5, b = 5
```

Câu hỏi

- Cho biết kết quả của các biểu thức sau:

1. x = 10; y = x++;
2. x = 10; y = ++x;
3.

```
int x = 5, y = 5, z;
x = ++x;
y = --y;
z = x++ + y--;
```

Câu hỏi

- Cho biết kết quả các đoạn code sau đây:

1)

```
int a =2, b = 3;
int m = a++ > b-- ? --a : b++;
cout << m+a+b;
```

2)

```
int a = 5, b = 9;
int m = a>b ? a++ : b++;
int n = a<b ? --a : --b;
cout << --n + (++m);
```

Câu hỏi

- Cho biết kết quả các đoạn code sau đây:

3)

```
#include<iostream.h>
int main()
{
        int a=5,b=4;
        cout<<a+++b<<endl;
        cout<<a+b<<endl;
        int c=5,d=4;
        cout<<++c+(++d)++<<endl;
        cout<<c+d<<endl;
        return 0;
}
```

Phép gán và biểu thức gán

- Phép gán (**assigment** operator) **"="** để thay đổi giá trị biến:
- **Cú pháp:** <tenBien> = <giatri>;

  <tenBien> = <tenBienkhac>;

  <tenBien> = <bieuThuc>;
- **Lưu ý:** phân biệt toán tử gán với khái niệm đẳng thức trong toán học.
- Biểu thức gán là biểu thức có dạng: **v = e**
  - Trong đó v là một biến, e là một biểu thức.
  - Giá trị của biểu thức gán là giá trị của e, kiểu của nó là kiểu của v.
- Nếu đặt dấu ; vào sau biểu thức gán thì ta được toán tử gán: **v = e;**

Phép gán và biểu thức gán

- Biểu thức gán có thể sử dụng trong các phép toán và các câu lệnh như các biểu thức khác.
- **Ví dụ 1:** a = b = 5;

  thì điều đó có nghĩa là gán giá trị của biểu thức: b = 5 cho biến a.

  Kết quả là b = 5 và a = 5.
- **Ví dụ 2:** Sau khi thực hiện câu lệnh

  z = (y = 2)*(x = 6);

  thì y có giá trị 2, x có giá trị 6 và z có giá trị 12.

Phép gán và biểu thức gán

- Phép toán kết hợp là phép gán cùng với phép toán, tác động lên chính biến được gán.

  += -= *= /= %=

  &= |= ^= <<= >>= // các toán tử trên bit

- **Ví dụ:**

  i += 2; //⇔ i = i + 2;

  a *= b+1; //⇔ a = a*(b + 1)

Ví dụ

```
void main()
{
    int a, b, c, d, e, thuong;
    a = 10;
    b = a;
    thuong = a / b;
    a = b = c = d = e = 156;
    e = 156;
    d = e;
    c = d;
    b = c;
    a = b;
}
```

Độ ưu tiên của các toán tử

- Khi thực hiện tính toán trong một biểu thức, phép toán có độ ưu tiên cao hơn sẽ thực hiện trước.

b = (a= 3)+2;

b = a= 3 + 2;

n = 18/4*4;

- Bảng sau cho biết độ ưu tiên phép toán (thứ tự giảm dần).
- Trừ phép gán và phép 1 toán hạng, **các phép cùng cấp sẽ ưu tiên trái hơn.**

Ví dụ:

```
n = 2 + 3 * 5;
      => n = 2 + (3 * 5);
a > 1 && b < 2
      => (a > 1) && (b < 2)
```

Độ ưu tiên của các toán tử.

Mức	Toán tử						Loại	Thứ tự
Cao nhất	::						Đơn hạng	Cả hai
	()	[]	->	.			Nhị hạng	Trái tới phải
	+ - -	++ -- --	! ~	* &	new delete	sizeof ()	Đơn hạng	Phải tới trái
	->*	.*					Nhị hạng	Trái tới phải
	*	/	%				Nhị hạng	Trái tới phải
	+	-					Nhị hạng	Trái tới phải
	<<	>>					Nhị hạng	Trái tới phải
	<	<=	>	>=			Nhị hạng	Trái tới phải
	==	!=					Nhị hạng	Trái tới phải
	&						Nhị hạng	Trái tới phải
	^						Nhị hạng	Trái tới phải
	|						Nhị hạng	Trái tới phải
	&&						Nhị hạng	Trái tới phải
	||						Nhị hạng	Trái tới phải
	? :						Tam hạng	Trái tới phải
	=	+= -=	*= /=	^= %=	&= |=	<<= >>=	Nhị hạng	Phải tới trái
Thấp nhất	,						Nhị hạng	Trái tới phải

Câu hỏi

- Cho biết kết quả các biểu thức sau đây:

1. (3.0/4 < 4.0/5) && (‘a’ < ‘b’)
2. (3/4 < 4/5) && (‘a’ < ‘b’)
3. !(48.5+2 < 50) || (2 > 4/2)
4. !(48.5+2 < 50) && (3> 4/2)

Ví dụ

❖ Dùng ngôn ngữ C viết biểu thức cho các mệnh đề

- x lớn hơn hay bằng 3: **x >= 3**
- a và b cùng dấu:

  ((a>0) && (b>0)) || ((a<0) && (b<0))

  (a>0 && b>0) || (a<0 && b<0)
- p bằng q bằng r : **(p == q) && (q == r) hoặc (p == q && q == r)**
- –5 < x < 5 : **(x > –5) && (x < 5) hoặc**

  (x > –5 && x < 5)

Lệnh và khối lệnh

- Câu lệnh (**statement**) : Là một chỉ thị trực tiếp, hoàn chỉnh nhằm ra lệnh cho máy tính thực hiện một số tác vụ nhất định nào đó. Trình biên dịch bỏ qua các khoảng trắng (hay tab hoặc xuống dòng) chen giữa lệnh.
- Ví dụ: các câu lệnh sau đây còn gọi là các lệnh đơn

```
int  a = 10;
x = 2*y;
b = (a >2 && b <=10);
```

Lệnh và khối lệnh

❖ Phân loại

- Câu lệnh đơn (**statement**): chỉ gồm một câu lệnh.
- Câu lệnh phức (khối lệnh – **block** statement): gồm nhiều câu lệnh đơn được bao bởi { và }
- Ví dụ: Các khối lệnh sau đây

```
{
        int x,y;
        x = 10;
        y = 20;
}
```

```
{
        float a,b,c;
        a = 1.5;
        b = 0.5;
        c = a+b;
}
```

```
int t;  // lệnh đơn
int x,y ;     // lệnh đơn
{
        t = x;
        x = y;
       y = t;
}
```

Lệnh Nhập/Xuất trong C và C++

- Lệnh xuất (**output**) dùng để in giá trị ra màn hình.
- Lệnh nhập (**input**) dùng để nhập giá trị từ màn phím vào cho biến trong chương trình.
- Lệnh nhập/xuất **trong C**
  - Lệnh nhập: **printf**
  - Lệnh xuất : **scanf**
  - Các lệnh nằm trong thư viện: **#include <stdio.h>** (yêu cầu phải khai báo thư viện để sử dụng các lệnh liên quan)
- Lệnh nhập/xuất **trong C++**
  - Lệnh nhập: **cin**
  - Lệnh xuất : **cout**
  - Các lệnh nằm trong thư viện: **#include <iostream>** (dùng cho môi trường Visual Studio)

    hoặc **#include <iostream.h>** (dùng cho môi trường C-Free, DEV-C)

    (yêu cầu phải khai báo thư viện để sử dụng các lệnh liên quan)

Lệnh Xuất: printf (trong C)

- Cú pháp : printf(<chuỗi định dạng>[, <đs1>, <đs2>, …]);
- Trong đó: <chuỗi định dạng> là cách trình bày thông tin xuất và được đặt trong cặp nháy kép " "
- Chuỗi định dạng thông thường chứa 3 thành phần:
  - Văn bản thường (literal text)
  - Ký tự điều khiển (escape sequence)
  - Đặc tả (conversion specifier)

Lệnh Xuất: printf (trong C)

❖ Văn bản thường (**literal text**)

- Được xuất ra màn hình y hệt như lúc gõ trong chuỗi định dạng.

Ví dụ:

- Xuất chuỗi "Hello World"

  ➔ printf("Hello"); printf("World");

  ➔ printf("Hello World");

- Xuất chuỗi "a + b"

  ➔ printf("a + b");

Lệnh Xuất: printf (trong C)

❖ Ký tự điều khiển (**escape sequence**)

- Gồm dấu \ và một ký tự như trong bảng sau:

Ký tự điều khiển	Ý nghĩa
\a	Tiếng chuông
\b	Lùi lại một bước
\n	Xuống dòng
\t	Dấu tab
\\	In dấu \
\?	In dấu ?
\"	In dấu "
\'	In dấu '
\r	Nhảy về đầu hàng, không xuống hàng
%%	In dấu %

Ví dụ:

```
printf("Hello\tHello");
printf("\n");
printf("World\n");
printf("ĐHSG");
printf("Khoa CNTT");
```

Kết quả in ra màn hình là:

```
Hello	Hello
World
ĐHSGKhoa CNTT
```

Ví dụ

```
#include <stdio.h>
 int main()
{
    printf("\a");
    printf("hello \n");
    printf("\tDong nay cach ra 1 tab ?\n");
    printf("\t\t\t\t\t\tCach ra nhieu tab qua \rVe dau dong thoi\n");
    printf("Dau \\ \nDau \'\nDau \" \nDau %%\n");

    // day la mot dong ghi chu va chi co tac dung chu thich khi viet code cho de hieu
     /*
      Day la mot doan ghi chu
      Doan ghi chu nay co 2 dong
     */
    return 0;
}
```

Lệnh Xuất: printf (trong C)

❖ Đặc tả (**conversion specifier**)

- Gồm dấu **%** và một ký tự.
- Xác định kiểu của biến/giá trị muốn xuất.
- Các **đối số** chính là các biến/giá trị muốn xuất, được liệt kê theo thứ tự cách nhau dấu phẩy.

Đặc tả	Ý nghĩa	Kiểu dữ liệu
%c	**Ký tự**	char
%d, %ld	**Số nguyên có dấu hệ 10**	int, short, long
%f, %lf	**Số thực** (VD 5.54 khi in sẽ ra 5.540000)	float, double
%e	Số thực (ký hiệu có số mũ)	float, double
%s	**Chuỗi ký tự**	char[], char*
%u	Số nguyên không dấu	unsigned int/short/long
%x	Số nguyên hex không dấu (hệ 16)	unsigned int
%o	Số nguyên bát phân không dấu (hệ 8)	unsigned int

Lệnh Xuất: printf (trong C)

- Ví dụ:

```
int a = 10, b = 20;
printf("%d", a);          ⇒Xuất ra 10
printf("%d", b);          ⇒ Xuất ra 20
printf("%d %d", a, b);    ⇒ Xuất ra 10 20
```

```
float x = 15.06;
printf("%f", x);          ⇒ Xuất ra 15.060000  //6 chữ số sau dấu chấm động
printf("%f", 1.0/3);      ⇒ Xuất ra 0.333333
```

Lệnh Xuất: printf (trong C)

❖ Phối hợp các thành phần

int a = 1, b = 2;

- Xuất 1 cong 2 bang 3 và xuống dòng.

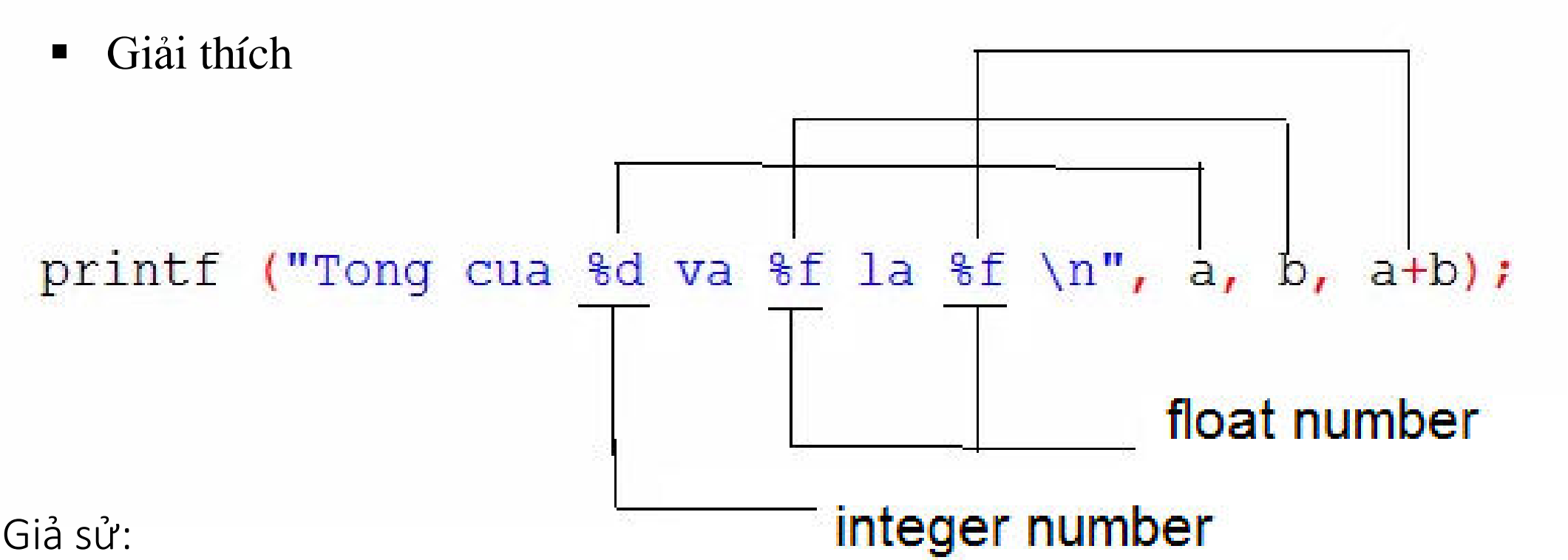
```
printf("%d", a);      // Xuất giá trị của biến a
printf(" cong  ");    // Xuất chuỗi "cong "
printf("%d", b);      // Xuất giá trị của biến b
printf(" bang  ");    // Xuất chuỗi " bang "
printf("%d", a + b);          // Xuất giá trị của a + b
printf("\n");                 // Xuất điều khiển xuống dòng \n
```

➔ printf("%d cong %d bang %d\n", a, b, a+b);

```
int a = 10, b = 20;
printf("gia tri cua a la: %d", a);          ⇒ Xuất ra: gia tri cua a la:10
printf("gia tri cua b la: %d", b);          ⇒ Xuất ra: gia tri cua b la:20
printf("tong cua %d va %d la %d", a, b, a+b);  ⇒ Xuất ra: tong cua 10 va 20 la 30
```

Lệnh Xuất: printf (trong C)

- Giải thích



Giả sử:

int a = 1;

float b =0.5;

Khi đó kết quả in ra màn hình là:

Tong cua 1 va 0.5 la 1.5 *(in ký tự xuống dòng)*

Ví dụ

```
#include <stdio.h>
 int main()
{
    int a = 12;
    float b = 13.5;
    char c = 'A';
    long d = 3454;
    char* s = "laptrinhC/C++"; // khai bao kieu chuoi
    printf("tong cua %d va %f la %f \n", a, b, a+b);
    printf("tich cua %d va %ld la %ld \n", a, d, a*d);
    printf("ky tu c la: %c \n", c);
    printf("chuoi s la: %s \n", s);
    printf("dinh dang so mu cua b la %e \n", b);
    printf("so he 16 va he 8 cua %d la %x va %o \n", a, a, a);
    printf("ma ASCII cua %c la %d\n", c, c);
    // system("pause"); // su dung de dung man hinh neu ban dung dev-C hoac Visual Studio
    return 0;
}
```

Lệnh Xuất: printf (trong C)

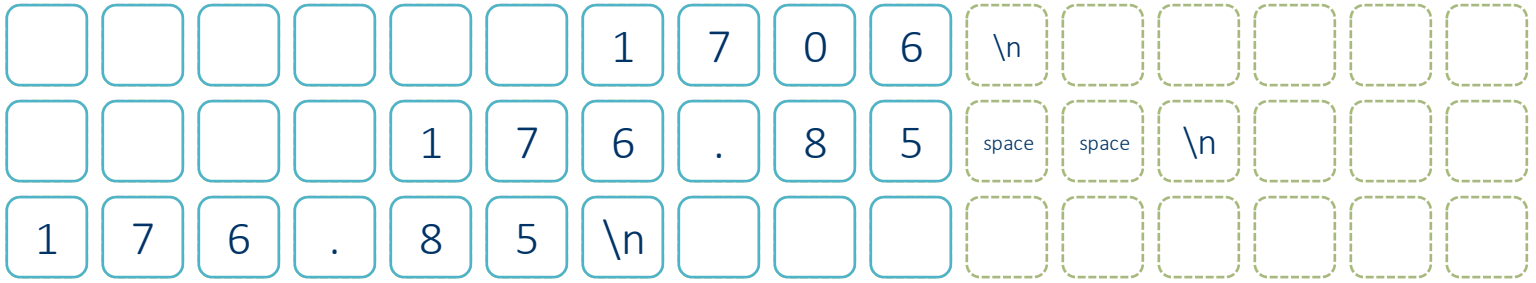
```
int a = 1706;
float x = 176.85;
printf("%10d", a);printf("\n");
printf("%10.2f \n", x);
printf("%.2f\n", x);
```

❖ Định dạng xuất

- Cú pháp
  - Định dạng xuất số nguyên: %**n**d, %**n**c, %**n**s

  (in số nguyên/ký tự/chuỗi có bề rộng là n)
  - Định dạng xuất số thực: %**n.k**f

  (in số thực có bề trọng là n với k chữ số sau dấu chấm động)



Ví dụ

```
#include <stdio.h>
 int main(){
    int a = 12;
    float b = 13.5;
    char c = 'Q';
    long d = 3454;
    char* s = "laptrinhC/C++"; // khai bao kieu chuoi
    printf("%6d %5.3f %.3f \n", a, b, a+b);
    printf("%-5d %5ld %5ld \n", a, d, a*d);
    printf("%5c \n", c);
    printf("%30s \n", s);
    // system("pause"); // su dung de dung man hinh neu ban dung dev-C or visual // studio
    return 0;
}
```

Lệnh nhập: scanf (trong C)

- Cú pháp: **scanf(<chuỗi định dạng>[, <đs1>, <đs2>, …]);**
- <chuỗi định dạng> giống định dạng xuất **nhưng chỉ có các đặc tả**.
- Các đối số là tên các biến sẽ chứa giá trị nhập và **được đặt sau dấu &.**

Ví dụ: cho a và b kiểu số nguyên

```
scanf("%d", &a);      // Nhập giá trị nguyên cho biến a
scanf("%d", &b);      // Nhập giá trị  nguyên cho biến b
```

➔ `scanf("%d %d", &a, &b);` // có thể nhập nhiều giá trị cùng lúc trong một lệnh

Lệnh Nhập: scanf (trong C)

- Ví dụ: Các câu lệnh sau đây **SAI**

```
scanf("%d", a);        // Thiếu dấu &
scanf("%d", &a, &b);   // Thiếu %d cho biến b
scanf("%f", &a);       // a là biến kiểu số nguyên
scanf("%9d", &a);      // không được định dạng
scanf("a = %d, b = %d", &a, &b); // chuỗi ngoài đặc tả
```

Ví dụ

```
#include <stdio.h>
 int main()
{
   int a;
   float b;

   printf("Nhap so nguyen a = ");
   scanf("%d", &a);

   printf("Nhap so thuc b = ");
   scanf("%f", &b);

   printf("a = %d \t b = %.3f", a, b);

   // system("pause"); // su dung de dung man hinh
   return 0;
}
```

Lệnh xuất/nhập trong C++

- Trong C++ thư viện nhập/xuất có tên là **<iostream.h>** (trong **Visual Studio** thì là **<iostream>**)
- Ngoài ra với C++ thì ta phải khai báo sử dụng thêm namespace std bằng cú pháp **using namespace std;**

```
#include <iostream>
using namespace std;
int main()
{
	// code here
	//system("pause"); // dung chuong trinh xem ket qua
	return 0;
}
```

Lệnh xuất: cout (trong C++)

- **cout**: là đối tượng xuất chuẩn, dùng để xuất một giá trị hoặc một biểu thức ra ngoài màn hình thiết bị hiển thị (*máy tính*).
- Một phát biểu xuất kết quả ra màn hình, bao gồm: **cout,** phép toán xuất **<<, *đối tượng được xuất, và ';'.***
- **Toán tử xuất <<** : standard output stream *// luồng ra.*
- Một đối tượng được xuất có thể là:
  - Số nguyên (số thực) hay biến nguyên (biến thực).
  - Kí tự (một hằng kí tự được đặt giữa cặp dấu ' ') hoặcbiến kiểu kí tự.
  - Thông điệp gồm nhiều kí tự (*chuỗi kí tự) được đặt* giữa cặp dấu " ".

Lệnh xuất: cout (trong C++)

- Nhiều phát biểu xuất có thể được nối lại thành một phát biểu xuất, khi đó cần lưu ý là trước mỗi đối tượng được xuất là một phép toán xuất và ngược lại.
- Các đối tượng được xuất ra sẽ liên tiếp nhau trên cùng một dòng.
- Nếu muốn một đối tượng được xuất ra trên một dòng mới thì gọi endl hay "\n" trong phát biểu xuất.

Ví dụ:

```
cout << "hello world";
int a = 5 ;  cout << a ;
cout << "Gia Tri 1" << "Gia tri 2" << "Gia tri 3"; // in ra nhiều output
cout << endl;  // endl : ký tự xuống dòng
cout << "\n";        // ký tự xuống dòng
int a = 1;
float b = 0.5, c;
cout <<" Tong cua " << a <<" va " <<" la: " << a+b;
```

Ví dụ

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Chao mung ban den voi mon hoc CSLT";
    cout << endl;
    cout << "Ban dang hoc series C++ can ban"<< endl;
    int a = 5;
    float b = 0.5;
    cout << "a = " <<a  << "    " << "b = " <<b << endl;
    //system("pause"); // dừng màn hình  xem kết quả
    return 0;
}
```

Lệnh nhập: cin (trong C++)

- **cin** là dòng nhập chuẩn, nhập/đọc dữ liệu được **gõ từ bàn phím**.
- Một phát biểu nhập dữ liệu từ bàn phím, bao gồm: **cin,** phép toán nhập **>>, *tên biến cần lưu giá trị nhập, và ';'.***
- Dạng tổng quát: cin >> var; (var là một biến nào đó).
- Phát biểu nhập có nhiều đối tượng:

  cin>>var1>>var2…>>varN;

- **Toán tử nhập** >> : standard input stream *// luồng vào*

Ví dụ: *int a ;* cin >> a;

cin >> variable1 >> variable2 >> variable3;

Lệnh nhập: cin (trong C++)

- Ví dụ:

```
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout << "nhap vao a" << endl ;
    cin >> a ;
    cout << "nhap vao b"<< endl;
    cin >> b;
    int tong = 0;
    tong = a+b;
    cout <<"tong = " << tong;
    // system("pause"); // dừng màn hình quan sát kết quả
    return 0;
}
```

Ép kiểu (type casting) trong C

- Trong một biểu thức, các toán hạng khác kiểu sẽ phải chuyển sang cùng kiểu để tính toán.
- **Ép kiểu (type casting)** là cách để chuyển đổi một biến từ kiểu dữ liệu này sang kiểu dữ liệu khác.
- Ví dụ, khi bạn muốn lưu trữ một giá trị *long* cho một số *số nguyên*, bạn phải ép kiểu *long* thành *int*.
- Có 2 loại ép kiểu: tường minh (**explicit**) và không tường minh (**implicit**)

Ép kiểu không tường minh

- Ép kiểu không tường minh:
  - Việc ***tự động chuyển kiểu được thực hiện từ*** toán hạng có kiểu "hẹp" sang kiểu "rộng" hơn.
  - Với phép gán, kết quả của biểu thức bên phải sẽ được chuyển thành *kiểu của biến bên trái.*
- Sự nâng cấp số nguyên là quá trình mà các giá trị của số nguyên *nhỏ hơn* **int** hoặc **unsigned int** chuyển đổi thành kiểu **int** hoặc **unsigned int**

Ví dụ:

```
int a = 1/2; // số nguyên a =1
float b = 1.0/2; // x =0.5
float x;
x = 3/4.0 + 2;   // x = ???
int y;
y = 3/4.0 + 2;   // y =??
```

```
include <stdio.h>
int main()
{
  int  i = 21;
  char c = 'c'; /* Gia tri ASCII la 99 */
  int tong;
  tong = i + c; // ky tu 'c' = 99 + gia tri cua i la 21
  printf("Gia tri cua tong la: %d\n", tong ); // gia tri 120
  printf("\n==========================\n");
  return 0 ;
}
```

Ép kiểu (type casting) trong C

- Ép kiểu tường minh: buộc kiểu của biểu thức chuyển sang kiểu khác. Chuyển đổi giá trị từ một kiểu này sang một kiểu khác sử dụng **toán tử ép kiểu** như cú pháp sau:

(ten-kieu) bieu_thuc;

Ví dụ:

```
float c = 35.81;
int b = (int)c + 1;  // ket qua 36
```

Ví dụ

```
#include <stdio.h>
int main()
{
   int sochia = 32, sobichia = 6;
   double kq;

   kq = (double) sochia / sobichia;   // check ket qua neu ko dung ep kieu (double) ???
   printf("Gia tri cua kq la: %f\n", kq );

   printf("\n============================\n");
   return 0;
}
```

Ép kiểu trong C++

- Dùng toán tử ép kiểu **static_cast**
- Cú pháp : **static_cast<tenKieu> (bieuThuc);**

Ví dụ:

```
char ch = 'A';
cout << static_cast<int>(ch) << endl; // in ra 65, not 'A'
```

```
int a = 10;
int b = 4;
float f = static_cast<float>(a) / b;
```

Các thư viện phổ biến trong C/C++

- Trong ngôn ngữ C

Tên	Chức năng
stdio.h	**Xuất, nhập với màn hình, file, bàn phím,…**
ctype.h	Kiểm tra các lớp ký tự (chữ số, chữ cái,…)
string.h	Xử lý chuỗi và bộ nhớ
memory.h	Cấp phát và quản lý bộ nhớ động
math.h	**Một số hàm toán học**
stdlib.h	Chuyển đổi dữ liệu số-chuỗi, cấp phát bộ nhớ,…
time.h	Các hàm về thời gian
…	…

Các thư viện phổ biến trong C/C++

❖ Các hàm trong thư việc toán học: #include <math.h>

- Toán tử 1 ngôi, 1 đầu vào : double, Trả kết quả: double
  - acos, asin, atan, cos, sin, …
  - exp, log, log10
  - sqrt
  - ceil, floor
  - abs, fabs
- Toán tử hai ngôi, 2 đầu vào: double, Trả kết quả: double
  - double pow(double x, double y)

Các thư viện phổ biến trong C/C++

- Ví dụ: dùng các hàm trong thư viện <math.h>

```
int x = 4, y = 3, z = -5;
float t = -1.2;
float kq1 = sqrt(x);
int kq2 = pow(x, y);
float kq3 = pow(x, 1/3);
float kq4 = pow(x, 1.0/3);
int kq5 = abs(z);
float kq6 = fabs(t);
```

Các thư viện phổ biến trong C/C++

- Trong C++

Tên	Chức năng
<iostream>	**Định nghĩa các đối tượng để đọc và viết ra các dòng tiêu chuẩn (*standard stream*) (xuất và nhập (dữ liệu) từ C++)**
<string>	Xử lý chuỗi
<limits>	Numeric limits
<complex>	Thư viện kiểu số phức
<algorithm>	Thư viện về các thuật toán : sắp xếp, tìm kiếm, min/max, …
…	…

Các ví dụ minh họa

❖ Viết chương trình C/C++ cho các yêu cầu sau:

Nhập năm sinh của một người và tính tuổi của người đó

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main()
{
	int NamSinh, Tuoi;
	printf("Nhap nam sinh: ");
	scanf("%d", &NamSinh);
	Tuoi = 2020 - NamSinh;
	printf("Tuoi cua ban la %d", Tuoi);
	return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
	int NamSinh, Tuoi;
	cout<<" Nhap nam sinh: ";
	cin>> NamSinh;
	cout <<endl;
	Tuoi = 2020 - NamSinh;
	cout<<"Tuoi cua ban la:" <<Tuoi;
	return 0;
}
```

Các ví dụ minh họa

```
#include<stdio.h>
#include <iostream>
#include<stdlib.h>
using namespace std;
int main()
{
	int a = 0, b = 0;
	int sum = 0, dif = 0;
	printf("Nhap vao so thu nhat: ");
	scanf("%d", &a);
	printf("Nhap vao so thu hai: ");
	scanf("%d", &b);
	sum = a + b;
	dif = a - b;
	printf("Tong cua hai so la %d\n", sum);
	printf("Hieu cua hai so la %d\n", dif);
	return 0;
}
```

Câu hỏi: chuyển đoạn code trên dùng các lệnh nhập/xuất với cin/cout

Các ví dụ minh họa

```
#include <stdio.h>
#include <limits.h>  // limits for interger
#include <float.h>   // limits for float

int main()
{
   printf("TYPE              %6s %20s %20s\n", "SIZE", "MIN VALUE", "MAX VALUE");
   printf("char:          %6ld byte %20d %20d\n", sizeof(char), CHAR_MIN, CHAR_MAX);
   printf("unsigned char: %6ld byte %20d %20d\n", sizeof(unsigned char), 0, UCHAR_MAX);
   printf("short:         %6ld byte %20d %20d\n", sizeof(short), SHRT_MIN, SHRT_MAX);
   printf("int:           %6ld byte %20d %20d\n", sizeof(int), INT_MIN, INT_MAX);
   printf("long:          %6ld byte %20ld %20ld\n", sizeof(long), LONG_MIN, LONG_MAX);
   printf("long long:     %6ld byte %20lld %20lld\n", sizeof(long long), LLONG_MIN, LLONG_MAX);
   printf("float:         %6ld byte %20e %20e\n", sizeof(float), FLT_MIN, FLT_MAX);
   printf("double:        %6ld byte %20e %20e\n", sizeof(double), DBL_MIN, DBL_MAX);
   printf("long double:   %6ld byte %20Le %20Le\n", sizeof(long double), LDBL_MIN, LDBL_MAX);
   return 0;
}
```

Các ví dụ minh họa

Câu hỏi: chuyển đoạn code trên dùng các lệnh nhập/xuất với cin/cout

```
#include <stdio.h>
#define AGE_MAX 150     // hang so
#define C 'a'           // hang ky tu
#define CITY_NAME "HANOI" // hang chuoi
 int main()
{
    printf("hang AGE_MAX = %d\n", AGE_MAX);
    printf("hang C = %c\n", C);
    printf("hang CITY_NAME = %s\n", CITY_NAME);

    // AGE_MAX = 10; // lenh nay sai vi hang khong the thay doi duoc gia tri
    return 0;
}
```

Các ví dụ minh họa

Câu hỏi: chuyển đoạn code trên dùng các lệnh nhập/xuất với cin/cout

- Viết chương trình tính chu vi và diện tích của hình tròn có bán kính r (r nhập từ bàn phím)

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159
int main()
{
   int r;
   float chuvi, dientich;
   printf("Nhap ban kinh r:");
   scanf("%d", &r);
   chuvi = 2*PI*r;
   dientich = PI*r*r;
   printf("Chu vi hinh tron la: %f\n", chuvi);
   printf("Dien tich hinh tron la: %f", dientich);
   return 0;
}
```

Các ví dụ minh họa

```
#include <stdio.h>
int main()
{          int a, b, tong, hieu ,tich;          float thuong;
          printf("nhap 2 so a, b: ");          scanf("%d%d", &a, &b);
          printf("so a la: %d\n", a);          printf("so b la: %d", b);
          tong = a+b;
          hieu = a-b;
          tich = a*b;
          thuong = (float)a/b;  // check ket qua neu ko dung ep kieu (float)
          printf("\n\ntong la %d\n", tong);
          printf("hieu la: %d\n", hieu);
          printf("tich la: %d\n", tich);
          printf("thuong la: %.3f\n", thuong);
          return 0 ;
}
```

Các ví dụ minh họa

- Nhập vào một số nguyên dương có 3 chữ số. Tính tổng các chữ số.
- Ví dụ : n = 120 , kết quả : 1+2+0 = 3

```
#include <iostream>

using namespace std;

int main()
{
   int n;
   cout <<"nhap vao so 3 chu so:";
   cin >> n;
   int donvi,chuc,tram;
   donvi = n%10;
   //cap nhat lai n bo di hang don vi
   n = n/10;
   chuc = n%10;
   n = n/10;
   tram = n%10;
   int tong = 0;
   tong = donvi+chuc+tram;
   cout<<"tong la:" << tong;
   return 0;
}
```

Các ví dụ minh họa

- Viết chương trình nhập vào một số a bất kỳ và in ra giá trị bình phương a^2 lập phương a^3 của a và giá trị a^4

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
   int n;
   int n2;
   float pp;
   cout<<"Nhap so nguyen duong n: ";
   cin >> n;
   n2 = n*n;
   cout <<"n^2 = "<<n2 <<endl;
   pp = pow(n,2);
   cout<<"dung ham power:n^2 =" << pp<<endl;
   return 0;
}
```

Tìm hiểu thêm (không bắt buộc)

- Toán tử sizeof
- Tham chiếu (reference) trong C
- Các hàm setw(), setprecision() dùng cho cout<<
- Phát sinh số ngẫu nhiên trong C và C++

Ôn Tập

1. Trình bày các kiểu dữ liệu cơ sở trong C và cho ví dụ.
2. Trình bày khái niệm về biến và cách sử dụng lệnh gán.
3. Trình bày khái niệm về biểu thức.

   Tại sao nên sử dụng cặp ngoặc đơn.
5. Trình bày cách định dạng nhập/xuất trong C.
6. Ép kiểu là gì? Cho ví dụ.

Câu hỏi

- Chỉ ra các lỗi sai trong các đoạn code sau đây và đưa ra cách khắc phục

1.

```
#include <iostream>
using namespace std;
int main()
{
	int a = 5
	int b = 2;
	float c;
	c == a/b;
	cout << c << endl;
	return 0;
}
```

Câu hỏi

- Chỉ ra các lỗi sai trong các đoạn code sau đây và đưa ra cách khắc phục

2.

```
#include <iostream>
using namespace std;
int main()
{
	cin>>a;
	cout << a << endl;
	return 0;
}
```

Câu hỏi

- Chỉ ra các lỗi sai trong các đoạn code sau đây và đưa ra cách khắc phục

3.

```
#include <iostream>
using namespace std;
int main()
{
	int a;
	cout << a << endl;
	int b = 10;
	a = a+b;
	cout << a << endl;
	return 0;
}
```

Câu hỏi

- Chỉ ra các lỗi sai trong các đoạn code sau đây và đưa ra cách khắc phục

4.

```
#include <iostream>
using namespace std;
int main()
{
	int a, b;
	int sum = a + b;
	cout << "Nhập 2 số cần tính tổng";
	cin >> a;
	cin >> b;
	cout << "Tổng là: " << sum
	return 0;
}
```

Câu hỏi

- Chỉ ra các lỗi sai trong các đoạn code sau đây và đưa ra cách khắc phục

5.

```
#include <iostream>
using namespace std;
int main()
{
	float a = 25;
	cout <<" Can bac hai cua 25 la:" sqrt(25);
	return 0;
}
```

Bài Tập

❖ Viết chương trình hoàn chỉnh cho các bài toán sau:

1. Viết chương trình nhập 2 số, đổi giá trị 2 số rồi in ra 2 số.

Ví dụ: nhập vào a = 5 ; b = 6 . Kết quả a = 6 và b = 5

2. Viết chương trình cho biết chữ số hàng nghìn, hàng trăm, hàng chục, hàng đơn vị của một số có 04 chữ số. Ví dụ khi nhập 1357 thì in ra:

Chữ số hàng nghìn: 1

Chữ số hàng trăm: 3

Chữ số hàng chục: 5

Chữ số hàng đơn vị: 7

Bài Tập

3. Viết chương trình thực hiện các yêu cầu sau (không dùng hàm chuyển đổi):

a. Nhập vào một kí tự và in ra mã ASCII tương ứng với kí tự đó.

b. Nhập vào một số nguyên (1 đến 255) và in ra kí tự có mã ASCII tương ứng.

4. Nhập vào số thực x, tính và in ra các giá trị y_1 :

$$y_1 = 4(x^2 + 10x\sqrt{x} + 3x + 1)$$

Bài Tập

5. **THCSLT1-01**

Cho tam giác ABC có số đo 3 cạnh là các số thực nhập từ bàn phím (giả thiết 3 cạnh vừa nhập là số đo 3 cạnh của tam giác). Hãy tính chu vi, diện tích và độ dài đường cao AH kẻ từ A của tam giác ABC.

Gợi ý:

$$h_a = 2\frac{\sqrt{p\,(p-a)\,(p-b)\,(p-c)}}{a}$$

Với **a, b, c** là độ dài các cạnh; **ha** là đường cao được kẻ từ đỉnh **A** xuống cạnh **BC**; **p** là nửa chu vi:

$$p = \frac{(a+b+c)}{2}$$

S (ABC) = ½ (ha * BC)

- Gợi ý bài 1 : Dùng hàm căn bậc 2 (sqrt() trong thư viện <math.h>)

```
#include <stdio.h>
#include <math.h>

int main ()
{

  printf("Can bac hai cua %lf bang %lf\n", 4.0, sqrt(4.0) );
  printf("Can bac hai cua %lf bang %lf\n", 5.0, sqrt(5.0) );

  return(0);
}
```

Bài Tập

6. Trong mặt phẳng tọa độ OXY cho 3 điểm A,B,C lần lượt có tọa độ là (x_a,y_a), (x_b,y_b), (x_c,y_c). Hãy tìm tọa độ trọng tâm, diện tích đường tròn nội tiếp, diện tích đường tròn ngoại tiếp của tam giác ABC.

Gợi ý: (xem slide sau)

Hint

- Xét tam giác ABC có độ dài các cạnh đối diện 3 góc A, B, C là a, b, c,

3 đỉnh có tọa độ là $(x_a, y_a), (x_b, y_b), (x_c, y_c)$

- Tâm đường tròn nội tiếp tam giác đó có tọa độ là:

$$\left(\frac{ax_a + bx_b + cx_c}{P}, \frac{ay_a + by_b + cy_c}{P}\right) = \frac{a}{P}(x_a, y_a) + \frac{b}{P}(x_b, y_b) + \frac{c}{P}(x_c, y_c).$$

ở đó $P = a + b + c$

- r là bán kính đường tròn nội tiếp $p = \frac{a+b+c}{2}$

$$r = \frac{2S}{a+b+c} = \frac{S}{p} = (p-a)\tan\frac{A}{2} = (p-b)\tan\frac{B}{2} = (p-c)\tan\frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}$$

Hint

Đặt

$$a = AB = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$b = BC = \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}$$

$$c = CA = \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}$$

$$p = \frac{a+b+c}{2}$$

$$M = S_{ABC} = \sqrt{p(p-a)(p-b)(p-c)}$$

Diện tích đương tròn nội tiếp

$$S1 = \pi(\frac{M}{p})^2$$

Diện tích đương tròn ngoại tiếp

$$S2 = \pi(\frac{abc}{4M})^2$$

Bài Tập

7. Viết chương trình nhập vào bán kính hình cầu, tính và in ra diện tích, thể tích của hình cầu đó.

Hint : $S = 4\pi R^2$ và $V = (4/3)\pi R^3$.

8. Viết chương trình nhập vào số giây từ 0 đến 86399, đổi số giây nhập vào thành dạng "gio:phut:giay".

   Ví dụ: **2:11:5**